Bounded Version Vectors using Mazurkiewicz Traces

Madhavan Mukund[1,2], Gautham Shenoy R[1], and S P Suresh[1,2]

[1] Chennai Mathematical Institute, India {madhavan,gautshen,spsuresh}@cmi.ac.in
[2] CNRS UMI 2000 ReLaX

Abstract. Version Vectors constitute an essential feature of distributed systems that enables the computing elements to keep track of causality between the events of the distributed systems. In this article, we study a variant named Bounded Version Vectors. We define the semantics of version vectors using the framework of Mazurkiewicz traces. We use these semantics along with the solution for the gossip problem to come up with a succinct bounded representation of version vectors in distributed environments where replicas communicate via pairwise synchronization.

1 Introduction

Brewer's CAP Theorem [GL02] states that distributed systems which are required to be highly available and partition tolerant cannot guarantee that the replicas are strongly consistent. In such situations, systems make do with weaker notions of consistency. A popular weaker notion of consistency is *eventual consistency*, which allows for the states of the replicas to diverge for a finite (but not necessarily bounded) period of time with a guarantee that the states will eventually converge. The properties of such *eventually consistent* systems have been studied in [SS05].

The replicas in an eventually consistent systems perform updates locally and propagate the updates to other replicas. Whenever a pair of replicas a and b participate in the exchange of their knowledge about the local states of other replicas in the system, they require a mechanism to decide which among a and b has the most up-to-date information about a third replica c. *Version vectors* play an important role in this decision-making process.

A typical implementation of version vectors uses integer counters. The size of the vector is equal to the number of replicas in the distributed system. Whenever a relevant event occurs locally at a replica a, the replica increments the counter corresponding to itself in its copy of the version vector.

In reactive systems, the replicas communicate with each other using various mechanisms. Two prominent ones are *epidemic propagation* and *pairwise synchronization*. In epidemic propagation, each replica sends its version vector to all other replicas. On receiving a copy from a remote replica, the local replica will update its own version vector by taking a pointwise maximum of the received version vector and its own version vector. In pairwise synchronization,

two replicas synchronize and simultaneously update both their version vectors by taking the pointwise maximum. In this article, we concentrate on pairwise synchronization as the communication mechanism.

It can be noted that in a reactive system, these counter values grow as the computations grow and hence version vectors in general are unbounded. However, we observe that given two replicas a and b with their states r_a and r_b, respectively, there can be only four possible relationships between them: r_a is more up to date than r_b (or vice-versa), or r_a and r_b cannot be compared since they have received concurrent updates or they are the same. Moreover, when these two replicas participate in *pairwise synchronization*, they jointly derive the new version vector that captures their state merger by locally comparing their respective version vectors. The bounded number of relations they capture, and the fact that they can be locally computed without the need for a global view, provides the motivation to seek the existence of a finite representation for version vectors. One such finite representation for version vectors was presented in [AAB04]. In this article we define the semantics of version vectors within the framework of Mazurkiewicz traces. Using these semantics we reduce the problem of comparing the integers in the version vectors to the problem of comparing appropriate trace-events. We shall finally adapt the solution for the gossip problem presented in [MS97] to provide a more succinct bounded representation for version vectors.

2 Version vectors

We restrict ourselves to eventually consistent distributed systems with N replicas $\{0, 1, \ldots, N-1\}$. Having initialized themselves by executing the initialization operation I, each of the replicas can perform one of the following operations:

- When a client requests an update operation to be performed, replica i performs the update locally. This is denoted by U^i. This changes the state of i.
- Periodically, replicas participate in *pairwise synchronization*, where they exchange their states and arrive at a common state to reflect this exchange and update of knowledge. Pairwise synchronization between i and j is denoted by S^{ij}.

An operation o is said to be a k-operation (for $k \in \{0, 1, \ldots, N-1\}$) if $o = U^k$ or $o = S^{ik}$ for some i. We also say that k participates in o.

A finite sequence of operations performed by the distributed system is known as a *run*. We denote a run by $\alpha = Io_1o_2 \ldots o_n$, where o_i is either an update operation or a synchronization operation.

The *version vector* of replica i at the end of a run α is a vector $V_i(\alpha)$ of N integer counters. The k^{th} entry of $V_i(\alpha)$, denoted by $V_i^k(\alpha)$, represents the most recent update of replica k that i is aware of, at the end of α. We say that a version vector $V_i(\alpha)$ *dominates* $V_j(\alpha)$ iff $\forall k \in [0 \ldots N-1] : \; V_i^k(\alpha) \geq V_j^k(\alpha)$

Whenever $V_i(\alpha)$ dominates $V_j(\alpha)$ and $V_i(\alpha) \neq V_j(\alpha)$, semantically it means that at the end of a run α, replica i is more up to date than j, which implies that i has received all the updates that j has received.

The join operation between the k^{th} components of $V_i(\alpha)$ and $V_j(\alpha)$ is defined as follows:

$$V_i^k(\alpha) \sqcup V_j^k(\alpha) = max(V_i^k(\alpha), V_j^k(\alpha)).$$

The semantics of the version vectors for the various operations are presented in Table 1.

Operation	**Semantics**
Initialization	$V_i^k(I) = 0$
Update	$V_i^k(\alpha \cdot U^a) = \begin{cases} V_i^k(\alpha) + 1 & \text{if } k = i = a \\ V_i^k(\alpha) & \text{otherwise} \end{cases}$
Synchronization	$V_i^k(\alpha \cdot S^{ab}) = \begin{cases} V_a^k(\alpha) \sqcup V_b^k(\alpha) & \text{if } i \in \{a, b\} \\ V_i^k(\alpha) & \text{otherwise} \end{cases}$

Table 1. Semantics of version vectors

3 Mazurkiewicz Traces and Version Vectors

Let $\alpha = Io_1o_2 \dots o_n$ be a run. We define $Evs(\alpha)$ to be $\{e_\perp, e_1, \dots, e_n\}$, the set of *events* at which the operations in α occur. We define $Ops(\alpha)$ to be $\{I, o_1, \dots, o_n\}$, the set of operations in α. An event e_i is said to be a k-event if o_i is a k-operation. We also say that k participates in the event e_i. For two k-events $e_i, e_j \in Evs(\alpha)$, we say that $e_i \leq_k e_j$ iff $i \leq j$. Note that $\leq_k$ is a total order for each k.

Definition 1 *Given a run $\alpha = Io_1 \dots o_n$, its* Mazurkiewicz trace *(referred to as* trace *henceforth), denoted $t(\alpha)$, is a triple $(\mathcal{E}, \leq, \lambda)$ such that:*

- $\mathcal{E} = Evs(\alpha)$
- $\lambda : \mathcal{E} \to Ops(\alpha)$ *such that* $\lambda(e_\perp) = I$ *and* $\lambda(e_i) = o_i$ *for* $i \in \{1, \dots, n\}$
- $\leq = (\bigcup_k \leq_k)^*$.

A triple $t = (\mathcal{E}, \leq, \lambda)$ is a trace *if $t = t(\alpha)$ for some run α.*

Note that for any two events e, f in a trace and any replica k, $e \leq_k f \Rightarrow e \leq f$. We say that $e \lessdot f$ if $e < f$ and there is no event g such that $e < g < f$. If $e \lessdot f$, we say that f is an *immediate successor* of e. Note that if $e \lessdot f$, there is a replica i such that both e and f are i-events.

Definition 2 *For a trace $t = (\mathcal{E}, \leq, \lambda)$ and $S \subseteq \mathcal{E}$, the subtrace of t induced by S is given by $t(S) = (S, \leq_S, \lambda_S)$ where:*

- $\leq_S = \leq \cap (S \times S)$
- $\lambda_S(e) = \lambda(e)$ for all $e \in S$

We shall now define the semantics of version vectors over traces.

Definition 3 *Let $t = (\mathcal{E}' \cup \{e\}, \leq, \lambda)$ and $t' = (\mathcal{E}', \leq', \lambda')$ be two traces such that $t(\mathcal{E}') = t'$. The version vector of replica i in trace t, denoted by $V_i(t)$ is inductively defined as follows.*

Case $e = e_\perp$ *: For all k, $V_i^k(t) = 0$.*
Case e **is a** j **event,** $j \neq i$ *: For all k, $V_i^k(t) = V_i^k(t')$.*
Case e **is a** U^i **event** *: For $k \neq i$, $V_i^k(t) = V_i^k(t')$. $V_i^i(t) = V_i^i(t') + 1$.*
Case e **is a** S^{ij} **event** *: For all k, $V_i^k(t) = V_j^k(t) = V_i^k(t') \sqcup V_j^k(t')$.*

In this representation, for $i \neq j$, $e_i \leq e_j$ implies that the event e_j occurs strictly after e_i and thus the replicas participating in $\lambda(e_j)$ know about the occurrence of the event e_i. This transfer of this knowledge can be traced along the path from e_i to e_j in the trace. We formalize this notion by defining *ideals*.

Definition 4 *Let $t = (\mathcal{E}, \leq, \lambda)$ be a trace of the run α. A subset $\mathcal{I} \subseteq \mathcal{E}$ is said to be an* ideal *iff $\forall e_i \in \mathcal{I}$ and $e_j \leq e_i, e_j \in \mathcal{I}$.*

Thus an ideal is a downward closed subset of $\mathcal{E}$ with respect to the partial order $\leq$. The following properties of ideals follow from the definition:

Proposition 5. *Let $t = (\mathcal{E}, \leq, \lambda)$ be a trace of some run α. Then the following are true:*

1. *$\mathcal{E}$ is an ideal.*
2. *If $e_i \in \mathcal{E}$, the set $\downarrow e_i = \{e_j | e_j \leq e_i\}$ is an ideal. (It is referred to as* **the ideal generated by** *e_i.)*
3. *Every ideal $\mathcal{I}$ is generated by its maximal elements. In other words $\mathcal{I} = \bigcup_{e_i \in \sup \mathcal{I}} \downarrow e_i$.*
4. *If $\mathcal{I}$ and $\mathcal{J}$ are ideals, so are $\mathcal{I} \cup \mathcal{J}$ and $\mathcal{I} \cap \mathcal{J}$.*

Our goal is to reduce the problem of comparing the integer values in the version vectors to the problem of comparing appropriate trace events. To that end, we define the following terminology which will be used in the remainder of the paper.

Definition 6 *Let $\mathcal{I}$ be an ideal and i, j and k be replicas.*

An i-event e is the maximal event *of i in $\mathcal{I}$, denoted by $max_i(\mathcal{I})$, if $f \leq e$ for all i-events f. If $t = (\mathcal{E}, \leq, \lambda)$ is a trace, then we use the notation $max_i(t)$ to denote $max_i(\mathcal{E})$.*

The view *of i in an ideal $\mathcal{I}$, denoted by $\partial_i(\mathcal{I})$, is $\downarrow max_i(\mathcal{I})$, the ideal generated by the maximal i-event in $\mathcal{I}$. For a trace $t = (\mathcal{E}, \leq, \lambda)$ and event $e \in \mathcal{E}$, we use the notations $\partial_i(t)$ and $\partial_i(e)$ to denote $\partial_i(\mathcal{E})$ and $\partial_i(\downarrow e)$, respectively.*

The latest information *that i has about j in $\mathcal{I}$, denoted by $latest_{i\to j}(\mathcal{I})$, is defined to be $max_j(\partial_i(\mathcal{I}))$, the maximal j-event in the view of i. If $t = (\mathcal{E}, \leq, \lambda)$ is a trace, then we use the notation $latest_{i\to j}(t)$ to denote $latest_{i\to j}(\mathcal{E})$.*

The maximal update event *of i in $\mathcal{I}$, denoted $update_i(\mathcal{I})$, is $max\{e \in \mathcal{I} : \lambda(e) = U^i\}$, the maximal i-event in $\mathcal{I}$ which is also an update event. If no such event exists then $update_i(\mathcal{I}) = e_\perp$ For a trace $t = (\mathcal{E}, \leq, \lambda)$ and event $e \in \mathcal{E}$, we use the notations $update_i(t)$ and $update_i(e)$ to denote $update_i(\mathcal{E})$ and $update_i(\downarrow e)$, respectively.*

The latest update information *that i has about k in $\mathcal{I}$, denoted $lu_i^k(\mathcal{I})$, is the maximal k-update event in the view of i in $\mathcal{I}$. Thus $lu_i^k(\mathcal{I}) \stackrel{\text{def}}{=} update_k(\partial_i(\mathcal{I})) = update_k(latest_{i\to k}(\mathcal{I}))$. If $t = (\mathcal{E}, \leq, \lambda)$ then we use $lu_i^k(t)$ to denote $lu_i^k(\mathcal{E})$.*

At this juncture, we shall prove the following lemma which we shall be alluding to in the rest of the paper.

Lemma 7 (Crossover point lemma). *Let $\mathcal{I} \subseteq \mathcal{E}$ be an ideal in a trace, and a be a replica. For events e_1 and e_2 such that $e_1 \in \partial_a(\mathcal{I})$, $e_2 \in \mathcal{I} \setminus \partial_a(\mathcal{I})$ and $e_1 < e_2$, there exists a replica c such that $e_1 \leq latest_{a\to c}(\mathcal{I}) \leq e_2$.*

Proof. Let P be a path from e_1 to e_2 such that for any two successive events e, f in P, $e \lessdot f$. Let e_3 be the maximal $\partial_a(\mathcal{I})$-event on this path. Let e_4 be the minimal element along P such that $e_4 \notin \partial_a(\mathcal{I})$. Clearly $e_1 \leq e_3$, $e_4 \leq e_2$ and $e_3 \lessdot e_4$. This means that there is a replica c such that both e_3 and e_4 are c-events.

Clearly $e_3 \leq latest_{a\to c}(\mathcal{I})$. Since $e_4 \notin \partial_a(\mathcal{I})$, $e_4 \not\leq latest_{a\to c}(\mathcal{I})$. But since e_4 is a c-event, $latest_{a\to c}(\mathcal{I}) < e_4$. Thus $e_1 \leq latest_{a\to c}(\mathcal{I}) \leq e_2$.

If e is an event in the trace t, the version vector of replica i at e, denoted by $V_i(e) = V_i(t(\downarrow e))$.

Note that in a run α, $V_i^k(\alpha)$ denotes the integer corresponding to latest k-update operation that replica i has received. If $t = t(\alpha)$ then $lu_i^k(t)$ denotes the latest k-update known to replica i. We formalise this intuition through the following theorem.

Theorem 8. *If α is a run, $t = t(\alpha)$, and i, k are replicas, then $V_i^k(\alpha) = V_i^k(t) = V_k^k(lu_i^k(t))$.*

Proof. We shall prove this result by induction on the length of the run α. Suppose α contains only one operation, I. Then, from the semantics of version vectors, for every i, k, $V_i^k(\alpha) = V_i^k(t) = 0$. Also, since for every i, k, $lu_i^k(t) = e_\perp$, and the trace corresponding to $\downarrow e_\perp$ is t itself, it follows that $V_k^k(lu_i^k(t)) = 0$.

Assume that for all runs whose length is smaller than n, the result holds, and let $\alpha = \alpha' \cdot o_{max}$ be a run of length n with trace $t = (\mathcal{E}, \leq, \lambda)$. Let e_{max} be a

maximal event in t with $\lambda(e_{max}) = o_{max}$. Define $\mathcal{E}' = \mathcal{E} \setminus \{e_{max}\}$ and $t' = t(\mathcal{E}')$. By induction hypothesis, for any i, k, $V_i^k(\alpha') = V_i^k(t') = V_k^k(lu_i^k(t'))$. There are the following cases to consider.

i **does not participate in** o_{max}**:** In this case $V_i^k(\alpha) = V_i^k(\alpha')$, $V_i^k(t) = V_i^k(t')$ and $lu_i^k(t) = lu_i^k(t')$. Thus $V_i^k(\alpha) = V_i^k(t) = V_k^k(lu_i^k(t))$.

$o_{max} = U^i$ **and** $i \neq k$**:** We argue as in the previous case.

$o_{max} = U^i$ **and** $k = i$**:** From the definition and induction hypothesis, $V_i^k(\alpha) = V_i^k(t) = V_k^k(lu_i^k(t')) + 1$. Now $lu_i^k(t) = e_{max}$ and if $t'' = t(\downarrow e_{max} \setminus \{e_{max}\})$ then $lu_i^k(t'') = lu_i^k(t')$. Thus $V_k^k(lu_i^k(t)) = V_k^k(lu_k^k(t)) = V_k^k(lu_k^k(t'')) + 1 = V_k^k(lu_i^k(t')) + 1 = V_i^k(t) = V_i^k(\alpha)$.

$o_{max} = S^{ij}$**:** Now $V_i^k(\alpha) = max(V_i^k(\alpha'), V_j^k(\alpha'))$, $V_i^k(t) = max(V_i^k(t'), V_j^k(t'))$. By induction hypothesis, this is the same as $max(V_k^k(lu_i^k(t')), V_k^k(lu_j^k(t')))$. Since e_{max} is not an update event, $lu_i^k(t) = max(lu_i^k(t'), lu_j^k(t'))$. It follows that $V_i^k(\alpha) = V_i^k(t) = V_k^k(lu_i^k(t))$.

Corollary 9 *If α is a run with trace t and a, b, k are replicas then*

- $V_a^k(\alpha) < V_b^k(\alpha)$ *iff* $lu_a^k(t) < lu_b^k(t)$.
- $V_a^k(\alpha) = V_b^k(\alpha)$ *iff* $lu_a^k(t) = lu_b^k(t)$.

Proof. $V_a^k(\alpha) \leq V_b^k(\alpha)$ iff $V_k^k(lu_a^k(t)) \leq V_k^k(lu_b^k(t))$ (from theorem 8) iff $lu_a^k(t) \leq lu_b^k(t)$ (using the fact that $lu_a^k(t)$ and $lu_b^k(t)$ are both k-update events, and using the semantics of update operations over trace events). The statements in the corollary follows from this.

With this we have reduced the problem of comparing integer values in the version vectors to the problem of comparing the corresponding update events in the trace of the run. We shall explore options to provide bounded representation for version vectors. We look at the gossip problem and its solution to achieve this.

4 Bounding the version vectors: Using gossip

Suppose a and b are replicas with version vectors $V_a(t)$ and $V_b(t)$ at the end of a trace t. We have already argued (in Corollary 9) that comparing $V_a^k(t)$ and $V_b^k(t)$ is equivalent to comparing $lu_a^k(t)$ with $lu_b^k(t)$. The following is an immediate consequence of the definition of latest update information.

Proposition 10. *Let $t = (\mathcal{E}, \leq, \lambda)$ be a trace and $a, b, k < N$. If $lu_a^k(t) \neq lu_b^k(t)$ then $lu_a^k(t) < lu_b^k(t)$ iff $latest_{a \to k}(t) < latest_{b \to k}(t)$.*

Thus the problem of comparing distinct k-update events corresponds to comparing the latest k-events in the views of the appropriate a and b. This latter problem is a special case of the *gossip problem* in which the number of replicas communicating with each other is restricted to two. A finite-state local solution to the gossip problem was provided in [MS97]. We adapt the solution to arrive at a bounded representation for version vectors.

Definition 11 *Let $t = (\mathcal{E}, \leq, \lambda)$ be an ideal and a be a replica.*

We define the primary information *of a in t, denoted by $Primary_a(t)$ to be the set of all the latest events in the view of a in t. Formally, $Primary_a(t) = \{latest_{a\to k}(t) \mid k < N\}$.*

We define the primary graph *of a in t to be $\mathcal{G}_a(t) = (Primary_a(t), \leq^t_a)$ where $\leq^t_a$ the partial order $\leq$ restricted to the events in $Primary_a(t)$.*

Our goal is to settle the question of comparing $latest_{a\to k}(t)$ and $latest_{b\to k}(t)$ using a finite amount of information that is also local to replicas a and b. If $latest_{a\to k}(t) \leq latest_{b\to k}(t)$ then $latest_{a\to k}(t)$ is in the view $\partial_a(t) \cap \partial_b(t)$. Thus, $latest_{a\to k}(t)$ is in the view generated by the maximal elements of $\partial_a(t) \cap \partial_b(t)$. We show the following nice property about these maximal elements.

Lemma 12 ([MS97]). *Let a and b be replicas and $t = (\mathcal{E}, \leq, \lambda)$ be a trace. If e is a maximal event in the ideal $\partial_a(t) \cap \partial_b(t)$ then $e \in Primary_a(t) \cap Primary_b(t)$.*

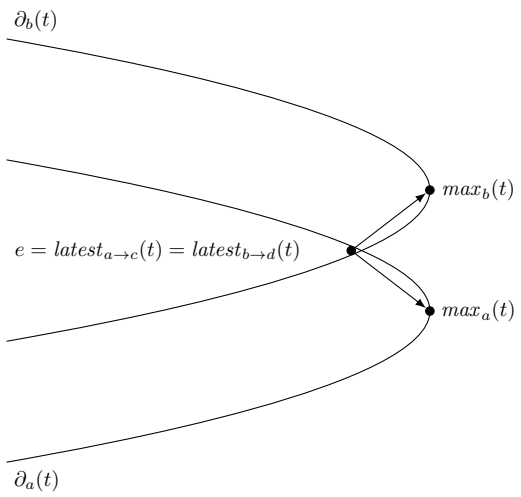

Fig. 1. Figure for lemma 12. e is a maximal event in $\partial_a(t) \cap \partial_b(t)$.

Proof. If $max_a(t) \in \partial_b(t)$ then, $\partial_a(t) \cap \partial_b(t) = \partial_a(t)$. Its sole maximal element is $max_a(t)$, which happens to be the same as $latest_{b\to a}(t)$, which is a member of $Primary_b(t)$ by definition. Thus $max_a(t) \in Primary_b(t)$. The case where $max_b(t) \in \partial_a(t)$ can be similarly handled.

Suppose it is the case that $max_a(t) \notin \partial_b(t)$ and $max_b(t) \notin \partial_a(t)$. Then $max_a(t) \in \partial_a(t) \setminus \partial_b(t)$. Similarly $max_b(t) \in \partial_b(t) \setminus \partial_a(t)$. From the crossover point lemma that there is c such that $e \leq latest_{a\to c}(t) \leq max_b(t)$. Since e is a

maximal event in $\partial_a(t)$, it follows that $e = latest_{a \to c}(t)$. Hence $e \in Primary_a(t)$. Similarly, $e = latest_{b \to d}(t)$ for some replica d and hence $e \in Primary_b(t)$. Thus any maximal $(\partial_a(t) \cap \partial_b(t))$-event e is in $Primary_a(t) \cap Primary_b(t)$.

This result gives us a way to compare $latest_{a \to k}(t)$ and $latest_{b \to k}(t)$ using the information present in the primary graphs $\mathcal{G}_a(t)$ and $\mathcal{G}_b(t)$. We record this through the following corollary.

Corollary 13 ([MS97]) *If $e = latest_{a \to k}(t)$ and $f = latest_{b \to k}(t)$ then $e \leq f$ iff $\exists g \in Primary_a(t) \cap Primary_b(t) : e \leq_a^t g$.*

If our purpose was to decide if $V_a(t)$ dominates $V_b(t)$ or if they are incomparable, then the information in $\mathcal{G}_a(t)$ and $\mathcal{G}_b(t)$ suffices since for any index k, whenever $latest_{a \to k}(t) \leq latest_{b \to k}(t)$ it is the case that $lu_a^k(t) \leq lu_b^k(t)$. However, if we also want to verify if the version vectors of the two replicas are the same, we have to show that for each k, $lu_a^k(t) = lu_b^k(t)$. For this we maintain the primary update information defined as follows.

Definition 14 *For any replica a and ideal $t = (\mathcal{E}, \leq, \lambda)$, the* primary update information*, denoted $PrimaryUpdate_a(t)$, is the indexed set $\{lu_a^k(t) \mid k \in [N]\}$.*

Thus for any replica a, the primary graph $\mathcal{G}_a(t)$ together with the primary update information $PrimaryUpdate_a(t)$ is an alternative representation of the version vector of $V_a(t)$. Our goal is to provide a bounded representation for this object. We do so by assigning labels to the events in the primary information and primary update information such that the labels come from a bounded set. Since the trace t can grow infinitely, finitely many labels implies reuse of labels at some point in time. However care must be taken to ensure that whenever two primary events $latest_{a \to i}(t)$ and $latest_{b \to j}(t)$ (or two primary update events $lu_a^i(t)$ and $lu_b^j(t)$) are assigned the same label, then they are indeed the same events. Otherwise the solution outlined above for comparing corresponding primary events would not work. Thus whenever we label a new S^{ab} event, we need to pick a label that is not currently in use for labelling any other primary event. We need to make this decision by looking at the local information of a and b. [MS97] shows that the information in the primary graphs $\mathcal{G}_a$ and $\mathcal{G}_b$ is not sufficient. Similarly when we label a new U^a event, we need to pick a label that is not currently in use for labelling any other primary update event. The information present in $PrimaryUpdate_a(t)$ is not sufficient for this purpose. So we define *secondary information*.

Definition 15 *Let $t = (\mathcal{E}, \leq, \lambda)$ be a trace and a, b, k be replicas.*
Define $latest_{a \to b \to k}(t) \stackrel{\text{def}}{=} latest_{b \to k}(\partial_a(t))$ and $lu_{a \to b}^k(t) \stackrel{\text{def}}{=} lu_b^k(\partial_a(t))$.
The secondary information *for a in t, denoted $Secondary_a(t)$, is defined as*

$$Secondary_a(t) = \{latest_{a \to b \to k}(t), lu_{a \to b}^k(t) \mid b, k < N\}$$

.

We now show that the secondary information of a replica a contains enough information for it to know if a certain a-event is in the primary information or primary update information of any other replica b.

Lemma 16. *Let a, b be replicas and $t = (\mathcal{E}, \leq, \lambda)$ be a trace. Then*

1. $latest_{b\to a}(t) \in Secondary_a(t)$.
2. $lu^a_b(t) \in Secondary_a(t)$.

Proof. 1. We need to show that $latest_{b\to a}(t) = latest_{a\to d\to a}(t)$ for some $d < N$. If $max_b(t) \in \partial_a(t)$ then $latest_{b\to a}(t) = latest_{b\to a}(\partial_a(t)) = latest_{a\to b\to a}(t)$. Suppose $max_b(t) \notin \partial_a(t)$. Since $latest_{b\to a}(t)$ is also an a-event, $latest_{b\to a}(t) \in \partial_a(t)$. By the crossover point lemma we can find c such that $latest_{b\to a}(t) \leq latest_{a\to c}(t) \leq max_b(t)$. Since $latest_{b\to a}(t)$ is the maximal a event in $\partial_b(t)$, we have $latest_{a\to c\to a}(t) = latest_{b\to a}(t)$. Thus $latest_{b\to a}(t) \in Secondary_a(t)$.
2. We need to show that there is c such that $lu^a_b(t) = lu^a_{a\to c}(t)$. If $max_b(t) \in \partial_a(t)$, then $lu^a_b(t) = lu^a_{a\to b}(t)$ and we are done.
Otherwise, we can appeal to the crossover lemma to show that there is a c such that $lu^a_b(t) \leq latest_{a\to c}(t)$ and $latest_{a\to c}(t) \leq max_b(\mathcal{E})$. It then follows that $lu^a_b(t) \leq lu^a_{a\to c}(t)$ and $lu^a_{a\to c}(t) \leq lu^a_b(t)$ and we are done.

Definition 17 *For a replica a and a trace t, the event version vector of a in t, denoted $EVV_a(t)$, is defined to be $(\mathcal{G}_a(t), PrimaryUpdate_a(t), Secondary_a(t))$.*

As the replicas participate in more operations, the trace t grows. We need to show that the event version vectors of the participating replicas can be reconstructed from their event version vectors before the operation.

Lemma 18. *Let $t = (\mathcal{E}, \leq, \lambda)$ and $t' = (\mathcal{E} \cup \{e_{new}\}, \leq', \lambda')$ be traces such that $t'(\mathcal{E}) = t$. Let $\lambda'(e_{new}) = o_{new}$. If replicas a, b participate in the operation o_{new} then $EVV_a(t')$ and $EVV_b(t')$ can be reconstructed from $EVV_a(t)$ and $EVV_b(t)$.*

Proof. We prove by induction on the size of t'. The base case is when $t' = (\{e_\perp\}, \leq', \lambda')$, in which case we have $latest_{a\to b}(t') = latest_{a\to b\to k}(t') = lu^b_a(t') = lu^k_{a\to b}(t') = e_\perp$ for any replicas a, b, k. And we have $e_\perp \leq^{t'}_a e_\perp$.

Suppose the result holds for all traces of size smaller than n. Suppose t' is of size n. We have the following two cases to consider.

$o_{new} = U^a$: From the definitions, for all $i \neq a$ and any k, $latest_{a\to i}(t') = latest_{a\to i}(t)$ and hence $latest_{a\to i\to k}(t') = latest_{a\to i\to k}(t)$, $lu^i_a(t') = lu^i_a(t)$ and $lu^k_{a\to i}(t') = lu^k_{a\to i}(t)$. Now $latest_{a\to a}(t') = latest_{a\to a\to a}(t') = lu^a_a(t') = lu^a_{a\to a}(t') = e_{new}$. For any two events $e, f \in Primary_a(t')$, $e \leq^{t'}_a f$ iff $e \leq^t_a f$ or $f = e_{new}$. Thus we have reconstructed $\mathcal{G}_a(t')$, $PrimaryUpdate_a(t')$ and $Secondary_a(t')$ thereby reconstructing $EVV_a(t')$.

$o_{new} = S^{ab}$: Let $i, j \in \{a, b\}$. From Corollary 13, for any replica $k \notin \{a, b\}$, it is possible to decide whether $latest_{a\to k}(t) \leq latest_{b\to k}(t)$ from the information in $\mathcal{G}_a(t)$ and $\mathcal{G}_b(t)$. Define w_k as follows:

$$w_k = \begin{cases} b & \text{if } latest_{a\to k}(t) \leq latest_{b\to k}(t) \\ a & \text{otherwise} \end{cases}$$

Now by definition, $latest_{i\to k}(t') = latest_{w_k\to k}(t)$ and $latest_{i\to j}(t') = e_{new}$. Similarly $lu^k_i(t') = lu^k_{w_k}(t)$ and $lu^j_i(t') = lu^j_j(t)$. Further, for any replica p, $latest_{i\to k\to p}(t') = latest_{w_k\to k\to p}(t)$ and $latest_{i\to j\to p}(t') = latest_{j\to p}(t')$. Similarly, $lu^p_{i\to k}(t') = lu^p_{w_k\to k}(t)$ and $lu^p_{i\to j}(t') = lu^p_j(t')$. If $e \in Primary_i(t')$ then $e \leq^{t'}_i e_{new}$. If $e, f \in Primary_i(t') \cap Primary_j(t)$ with $e \leq^t_j f$, then it is the case that $e \leq^{t'}_i f$. If $e, f \in Primary_i(t')$ but for $i \neq j$, $e \in Primary_i(t) \setminus Primary_j(t)$ and $f \in Primary_j(t) \setminus Primary_i(t)$, then e and f are incomparable in t, and hence are incomparable in t'. With this, we have reconstructed $\mathcal{G}_a(t')$, $\mathcal{G}_b(t')$, $PrimaryUpdate_a(t')$, $PrimaryUpdate_b(t')$, $Secondary_a(t')$ and $Secondary_b(t')$, thereby reconstructing $EVV_a(t')$ and $EVV_b(t')$.

Labelling

Let $\{\mathcal{L}_{ij} \mid i \leq j < N\}$ be a collection of mutually disjoint sets of labels, such that each $\mathcal{L}_{ij}$ is of size $2N + 1$. Let l_0 be a label not occurring in any of the $\mathcal{L}_{ij}$s. Let $\mathcal{L} = \bigcup_{i\leq j<N} \mathcal{L}_{ij} \cup \{l_0\}$. For a trace $t = (\mathcal{E}, \leq, \lambda)$, denote by $\mathcal{V}(t)$ the set of all events that appear in $EVV_i(t)$ for any $i < N$. We say that a labelling function $\rho : \mathcal{V}(t) \to \mathcal{L}$ is *legal* if the following conditions are satisfied:

- $\rho(e_\perp) = l_0$.
- If $e \in \mathcal{V}(t)$ is a U^a event then $\rho(e) \in \mathcal{L}_{aa}$.
- If $e \in \mathcal{V}(t)$ is a S^{ab} event then $\rho(e) \in \mathcal{L}_{ab}$.
- For any replicas a, b, i, j, if $e = latest_{a\to i}(t)$ (resp $lu^i_a(t)$) and $f = latest_{b\to j}(t)$ (resp $lu^j_b(t)$) and $\rho(e) = \rho(f)$ then $e = f$.

The following lemma shows how a legal labelling function for any trace can be extended to be a legal labelling function of its extension.

Lemma 19. *Let $t = (\mathcal{E}, \leq, \lambda)$ and $t' = (\mathcal{E} \cup \{e_{new}\}, \leq', \lambda')$ be traces with $t'(\mathcal{E}) = t$ and $\lambda'(e_{new}) = o_{new}$. Let a, b be replicas which participate in o_{new}. Let ρ be a valid labelling function over $\mathcal{V}(t)$. Then, using the information available in $EVV_a(t)$ and $EVV_b(t)$, one can construct a valid labelling function ρ' over $\mathcal{V}(t')$ such that for any $e \in \mathcal{V}(t) \cap \mathcal{V}(t')$, $\rho'(e) = \rho(e)$.*

Proof. We argue by induction on the size of t'. The base case is trivial when t' just consists of one event $e_\perp$ which is labelled by the unique label l_0.

Suppose the result holds for all traces of size smaller than n, and let t' be of size n. We have to consider the following two cases:

$o_{new} = U^a$: From lemma 16, for any replica i, the a-update event $lu^a_i(t) \in Secondary_a(t)$. Since there can be at most N such events and the set $\mathcal{L}_{aa}$ contains $2N + 1$ labels, we can always pick a label from $\mathcal{L}_{aa}$ for e_{new} which is different from any of $\rho(lu^a_i(t))$. Set $\rho'(e_{new})$ to be this new label and for every other $e \in \mathcal{V}(t) \cap \mathcal{V}(t')$, set $\rho'(e) = \rho(e)$. Since ρ is a legal labelling over $\mathcal{V}(t)$, ρ' is a legal labelling over $\mathcal{V}(t')$.

$o_{new} = S^{ab}$: From lemma 16, for any replicas $i, j \in \{a, b\}$, if $latest_{i \to j}(t)$ is an S^{ab} event, then $latest_{i \to j}(t) \in Secondary_a(t) \cap Secondary_b(t)$. Since there can be at most $2N$ such events and since the size of $\mathcal{L}_{ab}$ is $2N+1$, we can always pick a label from $\mathcal{L}_{ab}$ which is not assigned to any of these $latest_{i \to j}(t)$ events. Set $\rho'(e_{max})$ to that label. For every other $e \in \mathcal{V}(t) \cap \mathcal{V}(t')$, set $\rho'(e) = \rho(e)$. Since ρ is a legal labelling over t, ρ' is a legal labelling over t'.

Thus instead of storing the events of the traces in our event version vectors, if we store the corresponding labels which come from a finite set, we get a bounded representation for version vectors. Lemma 19 shows how to label new events such that the latest and the latest-update events are unambiguously identifiable. We now discuss the complexity of our solution.

Complexity of our Solution

For any replica i, the size of $EVV_i(t)$ is bounded by $O(N^2)$ since $Secondary_i(t)$ has at most N^2 events. We label the events from a finite collection of labels, where the size of each collection is bounded by $2N + 1$. Since there are at most N^2 such collections, the total number of labels in use is at most $2.N^3$. Thus we can store each label using $O(\log N)$ space. Since each replica stores at most N^2 labels, the space used by one replica is $O(N^2 \log N)$. Since there are N replicas, the overall space complexity is $O(N^3 \log N)$. This which compares favourably to the $O(N^4 \log N)$ space consumed by the solution proposed in [AAB04].

5 Conclusion and Summary

In this work, we have have defined the semantics of version vectors in the framework of Mazurkiewicz traces. We adapted the solution to the gossip problem as presented in [MS97] to arrive at a more succinct bounded representation for version vectors.

The bounded representation of a version vector presented in our work requires $O(N^3 \log N)$ space. It is not yet known whether this can be improved. Exploring that question is left for future work. Another interesting question to explore is whether version vectors have a bounded representation when the replicas communicate using message passing instead of pairwise synchronization. It is worth studying whether results on the gossip problem for message passing systems, studied in [MNS03], can be applied to this problem. We hope to investigate these issues in future work.

References

[AAB04] José Bacelar Almeida, Paulo Sérgio Almeida, and Carlos Baquero. Bounded version vectors. In *DISC*, pages 102–116, 2004.

[GL02] Seth Gilbert and Nancy A. Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2):51–59, 2002.

[MNS03] Madhavan Mukund, K. Narayan Kumar, and Milind A. Sohoni. Bounded time-stamping in message-passing systems. *Theoretical Computer Science*, 290(1):221–239, 2003.

[MS97] Madhavan Mukund and Milind A. Sohoni. Keeping track of the latest gossip in a distributed system. *Distributed Computing*, 10(3):137–148, 1997.

[SS05] Yasushi Saito and Marc Shapiro. Optimistic replication. *ACM Computing Surveys*, 37(1):42–81, 2005.